

Amendments to the Claims:

The listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

- Al
- 1). (Original) A method, comprising:
 optimizing an implementation of a programming language, comprising;
 analyzing one or more values computed by a program written in the programming language, wherein analyzing one or more values comprises;
 representing each bit within a value of the one or more values as an abstract element of a lattice having a set of abstract elements including 0_A , 1_A , \perp_A and T_A , wherein the lattice is an abstraction of a concrete domain containing 0, 1, and \perp ;
 analyzing one or more output bits that are produced by an operation in terms of one or more input bits that are input to the operation; and
 analyzing the input bits that are input to the operation in terms of the output bits that are produced by the operation.
 - 2). (Original) The method of claim 1, wherein optimizing further comprises:
 applying a forward abstract semantic to the abstract element; and
 applying a backward abstract semantic to the abstract element;
 wherein the forward abstract semantic is an approximation of a forward concrete semantic including AND, OR, and NOT; and
 wherein the backward abstract semantic is an approximation of a backward concrete semantic including AND^{-1} , OR^{-1} , and NOT^{-1} .
 - 3). (Original) The method of claim 2, further comprising:
 identifying the values within the program as partially constant values.
 - 4). (Original) The method of claim 3, wherein the backward abstract semantic is for a complex boolean function including $LEFT^{-1}$, $URIGHT^{-1}$, $JOIN^{-1}$, $MEET^{-1}$, LE^{-1} and $SRIGHT^{-1}$, and wherein the forward abstract semantic is for the complex boolean function including LEFT, URIGHT, JOIN, MEET, LE, and SRIGHT.
 - 5). (Original) The method of claim 4, wherein the program is represented in an intermediate language.

- AM
- 6). (Original) The method of claim 5, wherein the implementation is a compiler for the programming language.
 - 7). (Original) The method of claim 5, wherein the implementation is a computer aided design compiler for the programming language.
 - 8). (Original) A computer-readable medium having stored thereon a plurality of instructions, said plurality of instructions when executed by a computer, cause said computer to perform:
optimizing an implementation of a programming language, comprising;
analyzing one or more values computed by a program written in the programming language, wherein analyzing one or more values comprises;
representing each bit within a value of the one or more values as an abstract element of a lattice having a set of abstract elements including 0_A , 1_A , \perp_A and T_A , wherein the lattice is an abstraction of a concrete domain containing 0, 1, and \perp ;
analyzing one or more output bits that are produced by an operation in terms of one or more input bits that are input to the operation; and
analyzing the input bits that are input to the operation in terms of the output bits that are produced by the operation.
 - 9). (Original) The computer-readable medium of claim 8 having stored thereon additional instructions, said additional instructions when executed by a computer for optimizing, cause said computer to further perform:
applying a forward abstract semantic to the abstract element; and
applying a backward abstract semantic to the abstract element;
wherein the forward abstract semantic is an approximation of a forward concrete semantic including AND, OR, and NOT; and
wherein the backward abstract semantic is an approximation of a backward concrete semantic including AND^{-1} , OR^{-1} , and NOT^{-1} .
 - 10). (Original) The computer-readable medium of claim 9 having stored thereon additional instructions, said additional instructions when executed by a computer, cause said computer to further perform:
identifying the values within the program as partially constant values.

- 11). (Original) The computer-readable medium of claim 10, wherein the backward abstract semantic is for a complex boolean function including $LEFT^{-1}$, $URIGHT^{-1}$, $JOIN^{-1}$, $MEET^{-1}$, LE^{-1} and $SRIGHT^{-1}$, and wherein the forward abstract semantic is for the complex boolean function including $LEFT$, $URIGHT$, $JOIN$, $MEET$, LE , and $SRIGHT$.
- 12). (Original) The computer-readable medium of claim 11, wherein the program is represented in an intermediate language.
- 13). (Original) The computer-readable medium of claim 11, wherein the implementation is a computer aided design compiler for the programming language.
- 14). (Original) A system, comprising:
a processor;
memory connected to the processor storing instructions for bidirectional bitwise constant propagation by abstract interpretation executed by the processor;
storage connected to the processor that stores a software program having a plurality of separately compilable routines,
wherein the processor optimizes an implementation of a programming language, by
analyzing one or more values computed by a program written in the programming language, wherein analyzing one or more values comprises;
representing each bit within a value of the one or more values as an abstract element of a lattice having a set of abstract elements including 0_A , 1_A , \perp_A and T_A , wherein the lattice is an abstraction of a concrete domain containing 0, 1, and \perp ;
analyzing one or more output bits that are produced by an operation in terms of one or more input bits that are input to the operation; and
analyzing the input bits that are input to the operation in terms of the output bits that are produced by the operation.
- 15). (Original) The system of claim 14, wherein the processor further optimizes by applying a forward abstract semantic to the abstract element; and
applying a backward abstract semantic to the abstract element;
wherein the forward abstract semantic is an approximation of a forward concrete semantic including AND, OR, and NOT; and

wherein the backward abstract semantic is an approximation of a backward concrete semantic including AND^{-1} , OR^{-1} , and NOT^{-1} .

- 16). (Original) The system of claim 15, wherein the processor identifies the values within the program as partially constant values.
- 17). (Original) The system of claim 16, wherein the backward abstract semantic is for a complex boolean function including LEFT^{-1} , URIGHT^{-1} , JOIN^{-1} , MEET^{-1} , LE^{-1} and SRIGHT^{-1} , and wherein the forward abstract semantic is for the complex boolean function including LEFT , URIGHT , JOIN , MEET , LE , and SRIGHT .
- 18). (Original) The system of claim 17, wherein the program is represented in an intermediate language.
- 19). (Original) The system of claim 18, wherein the implementation is a compiler for the programming language.
- 20). (Original) The system of claim 19, wherein the implementation is a computer aided design compiler for the programming language.
- 21). (Original) A system, comprising:
means for optimizing an implementation of a programming language, comprising;
means for analyzing one or more values computed by a program written in the programming language, wherein analyzing one or more values comprises;
means for representing each bit within a value of the one or more values as an abstract element of a lattice having a set of abstract elements including 0_A , 1_A , \perp_A and T_A , wherein the lattice is an abstraction of a concrete domain containing 0, 1, and \perp ;
means for analyzing one or more output bits that are produced by an operation in terms of one or more input bits that are input to the operation; and
means for analyzing the input bits that are input to the operation in terms of the output bits that are produced by the operation.
- 22). (Original) The system of claim 21, wherein the means for optimizing further comprises:

means for applying a forward abstract semantic to the abstract element; and
means for applying a backward abstract semantic to the abstract element;
wherein the forward abstract semantic is an approximation of a forward concrete semantic
including AND, OR, and NOT; and
wherein the backward abstract semantic is an approximation of a backward concrete semantic
including AND^{-1} , OR^{-1} , and NOT^{-1} .

- As
end
- 23). (Original) The system of claim 22, further comprising:
means for identifying the values within the program as partially constant values.
- 24). (Original) The system of claim 23, wherein the backward abstract semantic is for a complex boolean function including $LEFT^{-1}$, $URIGHT^{-1}$, $JOIN^{-1}$, $MEET^{-1}$, LE^{-1} and $SRIGHT^{-1}$, and wherein the forward abstract semantic is for the complex boolean function including LEFT, URIGHT, JOIN, MEET, LE, and SRIGHT.
- 25). (Original) The system of claim 24, wherein the program is represented in an intermediate language.
- 26). (Original) The system of claim 25, wherein the implementation is a compiler for the programming language.
- 27). (Original) The system of claim 26, wherein the implementation is a computer aided design compiler for the programming language.
-